`aligned-textgrid`: Lightweight access to structured phonetic data.

Josef Fruehwald
University of Kentucky
Department of Linguistics
josef.fruehwald@uky.edu

Christian Brickhouse
Stanford University
Department of Linguistics
brickhouse@stanford.edu

1 Introduction

Praat [Boersma and Weenink, 2024] is free and open source software most commonly used in linguistic research for acoustic analysis and audio annotation. Its TextGrid format allows for flexible, time-stamped, point and interval annotations of audio that can be easily edited in the Praat GUI and saved as plain text files.

In addition to manual annotation, a number of commonly used forced-alignment programs also generate highly structured data in the form of TextGrids, including the FAVE suite [Rosenfelder et al., 2022], DARLA [Reddy and Stanford, 2015], and the Montreal Forced Aligner [McAuliffe et al., 2017b]. While there are, additionally, a number of software tools specifically for parsing TextGrid files (in R, Mahr [2020], in python, Gorman [2019], Taubert [2023], Mahrt [2023], among many others), these tools don't focus on the linguistic *structure* represented in the TextGrid output of forced aligners. PolyglotDB [McAuliffe et al., 2017a] is a database framework that enriches TextGrids with an annotation graph, but its use case is analysis of larger-scale speech corpora. The goal of aligned-textgrid is to provide lightweight, scriptable access to the structured data produced by forced-aligners. The library is written in python, and currently available on the Python Package Index.

2 Hierarchy and Precedence in TextGrids

Figure 1 illustrates the typical output of a forced-aligner. It has two tiers: a word tier and a phone tier. Within the TextGrid format, each annotation is defined exclusively by i) its tier membership, ii) onset time, iii) offset time, and iv) label.

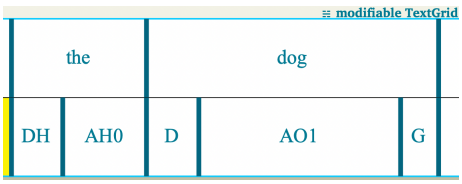

Figure 1: Word and Phone TextGrid

However, there are implicit linguistic relationships that are *not* well represented in the TextGrid. For example

1. Hierarchical Information
   a. Every phone is contained within one (and only one) word.
   b. Every word contains at least one phone.
2. Precedence Information
   a. The phone `AO1` follows the phone `D`.
   b. The phone `AO1` precedes the phone `G`
3. Hybrid Hierarchical & Precedence Information
   a. `G` is the last phone within the word dog.
   b. The way that `AO1` follows `D` (within a word) is not the same as the way `D` follows `AH0` (they cross words).

Quick and easy access to this hierarchical and precedence information may be of interest to researchers for a number of reasons. For example, in many North American varieties of English, the vowel quality of /ay/ is raised before voiceless consonants [Davis and Berkson, 2021], but *not* if a word boundary intervenes. In trying to programmatically identify examples of /ay/ that may be susceptible to raising, a researcher could not simply depend on examining the `i+1` interval, but rather would have to access information about the word tier above.

3 The aligned-textgrid approach

The overarching design goal of aligned-textgrid is to provide users named accessors to linguistically relevant data within a TextGrid from nearly any starting point, although, more conventional numerical indexing is also implemented.

3.1 Precedence Relationships

For example, the susceptibility of a phone to American Raising could be checked like so.

```
for phone in phone_tier:
  if phone.label == "AY1" \
   and phone.fol.label in voiceless_list:
    return phone
```

The `.fol` accessor returns the following interval instance. In this sense, precedence relationships are defined as a form of linked-list, except they are curtailed at the edge of the containing interval from the tier above. If the phone interval were the final phone in

Proceedings of the Society for Computation in Linguistics (SCiL) 2024, pages 329-330.
Irvine, California, June 27-29, 2024

accessor	description
`.fol`	return the following interval
`.prev`	return the previous interval

Table 1: Precedence Accessors

accessor	description
`.within`	return the containing interval
`.contains`	return a list of contained intervals

Table 2: Hierarchy Accessors

the word “my”, `phone.fol.label` would return `"#"`.

3.2 Hierarchical Relationships

Researchers may also want ready access to the hierarchical relationships as well. For example, /ay/ in some lexical items such as *spider* undergo exceptional raising [Fruehwald, 2008]. Still within a single iteration over the phone tier, the containing word can be accessed like so:

```
for phone in phone_tier:
  if phone.label == "AY1" \
   and phone.within.label == "spider":
    return phone
```

The `.within` accessor returns the interval from the tier above that the phone is contained within.

Hierarchical information in TextGrids is usually inferred from the linear order of tiers leading to unexpected behavior if researchers order tiers in unconventional ways. aligned-textgrid defines hierarchical relationships within SequenceTier classes, rather than relying on the order of the underlying text grid, to ensure that Words always contain Phones even if the phone tier is above the word tier.

3.3 Hybrid Relationships

Although `.prev` and `.fol` are curtailed by word boundaries, there are often reasons to search across word boundaries. For example, American /t/ flapping can happen at the end of a word when the following vowel is unstressed.

```
for phone in phone_tier:
  if phone.label == "T" \
   and phone.fol.label == "#" \
   and "0" in phone.within.fol.first.label:
    return phone
```

This example demonstrates how a textgrid can be navigated with the chaining of accessors. First `.within` returns the word interval. Then `.fol` returns the following word interval. Finally `.first` returns the first phone within the word.

accessor	description
`.first`	return the first contained interval
`.last`	return the last contained interval

Table 3: Hybrid Accessors

4 Flexibility

aligned-textgrid comes with default `Word` and `Phone` classes implemented, but also allows for on-the-fly definition of custom tier hierarchies via a `custom_classes()` class factory function, allowing for larger phrase annotation, or intermediate syllabic or prosodic annotations to be handled. The library is not restricted to work with only English or CMU phone labels.

References

Paul Boersma and David Weenink. 2024. Praat: doing phonetics by computer (version 5.0.35) [computer program] version 6.4.04.

Stuart Davis and Kelly Berkson. 2021. American raising: An introduction. *The Publication of the American Dialect Society*, 106(1):1–12.

Josef Fruehwald. 2008. The spread of raising : Opacity , lexicalization , and diffusion. *Penn Working Papers in Linguistics*, 14(2):83–92.

Kyle Gorman. 2019. *TextGrid: Praat TextGrid manipulation.*

Tristan Mahr. 2020. *readtextgrid: Read in a 'Praat' 'TextGrid' File*. R package version 0.1.1.

Tim Mahrt. 2023. *praatio: A library for working with praat, textgrids, time aligned audio transcripts, and audio files.*

Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. 2017a. Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi. In *Proc. Interspeech 2017*, pages 498–502.

Michael McAuliffe, Elias Stengel-Eskin, Michaela Socolof, and Morgan Sonderegger. 2017b. Polyglot and speech corpus tools: A system for representing, integrating, and querying speech corpora. In *Proc. Interspeech 2017*, pages 3887–3891. ISCA.

Sravana Reddy and James Stanford. 2015. A web application for automated dialect analysis. page 71–75, Denver, Colorado. Association for Computational Linguistics.

Ingrid Rosenfelder, Josef Fruehwald, Christian Brickhouse, Keelan Evanini, Scott Seyfarth, Kyle Gorman, Hilary Prichard, and Jiahong Yuan. 2022. *FAVE (Forced Alignment and Vowel Extraction) Program Suite v2.0.0*.

Stefan Taubert. 2023. textgrid-tools.